Remarks

The Applicants believe that this amendment places the subject application in better condition for allowance and in so doing introduces no new issues. Therefore, entry of this Amendment, reconsideration of the application, and allowance of all claims pending herein is respectfully requested.

Claims 1-32 were originally presented in the subject application. By the foregoing amendment, claims 1-7, 9-10, 13, 15, 17, 19, 21-24, 26, and 32 have been amended to address the Examiner's indicated informalities and to more particularly point out and distinctly claim the inventive material of the subject invention. In addition, claims 25 and 29-31 are canceled without prejudice or disclaimer. Claims 1-24, 26-28, and 32 remain in this case.

Applicants have endeavored to address the Examiner's concerns by adding clarifying language to various claims. The Examiner's concerns are addressed separately below in the order raised in the outstanding Office action.

No new matter has been added.

Rejections under 35 U.S.C. §112

Claims 1, 9, 10, 15, 26, 29, and 32 stand rejected 35 U.S.C. §112 in view of language reciting "inserting… into" an address. The Office action states that "in claim 1 as well as claims 9, 10, 15, 26, 29, and 32, '[I]nserting… into' an address is unclear because it is not clear… how an address field receives an instruction, as distinct from subsequent recitation of 'inserting at'…." Applicants submit that the terms 'inserting into' and 'inserting at' have been used interchangeably. However, in order to address the Examiner's concerns and to enhance clarity, the claims at issue have been amended to all recite 'inserting at' an address.

Claim 26 has been similarly rejected due to the recitation of "having a first vector address executable". The Office action states that "[n]o disclosure can be found that enables executing of addresses". Applicants respectfully submit that the language in question refers simply to the conventional flow of program execution in which a program counter (PC) increments from address to address as a program is executed. Such program execution is discussed, for example,

in the specification on lines 3-8 of page 15, which states that:

> [a]fter executing the first inserted instruction 20 located at vector address 12, execution falls through to the next instruction address, which in the embodiment shown, is at FIQ vector address 14. Branch instruction 22 previously inserted at this address 14 then instructs the processor to branch to common dispatcher 24.

As such, this language should be viewed as enabled and as particularly pointing out and distinctly claiming the subject matter which Applicants regard as the invention. However, in order to fully comply with the Examiner's concerns, the language at issue has been deleted from all claims where it originally appeared.

Claim 1 similarly stands rejected, with the Office action stating that with respect to "[t]he 'address preceding' [it] is unclear what precedence (by priority, numerical, etc.) is intended."

Applicants respectfully submit that this language simply refers to well-known software execution functionality in which programs necessarily execute sequentially, e.g., with the program counter advancing (provided there are no branches or interrupts, etc.) from a particularly numbered address to the next higher (e.g., even) numbered address. See, for example, lines 15-20 of page 14, which state:

> [a]s shown, the IRQ vector address 12 precedes the FIQ vector address 14 in conventional ARM® exception vector tables. As such, absent a branch instruction or other exception, such as a reset, instruction execution will generally advance from address 12 to address 14.

Nevertheless, to ensure that the Examiner's concerns are fully alleviated, and to provide enhanced clarity, Applicants have amended this language in various claims to recite "operationally sequential placement" or "operationally preceding" vector addresses. The claim, as so amended, should thus be viewed as specifically pointing out and distinctly claiming the subject matter which the Applicants regard as their invention.

The Office action further states that in "claim 32, 'adapted to' is unclear because it does not make clear what manner of adaptation is intended."

The language at issue has been deleted, and Applicants therefore submit that the claims as amended now particularly point out and distinctly claim the present invention.

Rejections under 35 U.S.C. §§ 102 & 103:

Claims 1-32 stand rejected under 35 U.S.C. §§102 & 103 in view of "Bhagat" (U.S. Patent Application Publication No. 2002/0016880) alone or in combination with 'Note 25' (ARM application note 25: Exception handling on the ARM (including thumb-aware processors)) or further in view of 'Note 31' (ARM application note 31: Using embeddedICE). This rejection, to the extent deemed relevant to the amended claims presented herewith, is respectfully traversed.

The independent claims each refer to merging interrupt streams of at least two discrete types at the processor level, e.g., merging the interrupt streams once they are effectively received at the interrupt vector addresses of the processor. To clarify this aspect, the independent claims that did not originally state it, have been amended to explicitly recite a "processor… having… first type and second type interrupt vector addresses".

Applicants believe this characteristic was implicit in the subject claims as originally presented, by virtue, for example, of claim 26, which originally recited "a processor having… a first vector address executable upon receipt of an interrupt request of the first type, and a second vector address executable upon receipt of an interrupt request of the second type, the first vector address preceding the second vector address in the vector table", such that no new search is required and no new matter has been added. Further support for this Amendment may be found in, for example, Fig. 2 and on lines 15-17 of the specification, which state that "[a]s shown, the IRQ vector address 12 precedes the FIQ vector address 14 in conventional ARM® exception vector tables."

This characteristic is neither disclosed nor suggested by the prior art cited by the Examiner. Specifically, regarding independent claims 1, 9, 15, 17, 19, 22, 23, 24, 26 and 32, Applicants submit that the cited references fail to disclose processors having such a merged interrupt stream. In particular, as discussed below, while the Bhagat patent appears to disclose an interrupt controller 106 that may merge interrupts from various sources (108-126), it does not do so at the chip or processor level. Rather, Bhagat's interrupt controller 106 is disposed upstream of ARM® processor 102, and sends discrete (i.e., unmerged) FIQ and IRQ interrupt streams 128 and 130 to the processor. The processor 102 then handles these interrupt streams in a

conventional manner. Advantageously, the claimed invention serves to nominally eliminate the occurrence of errors due to race conditions between discrete FIQ and IRQ interrupts during handling by the processor, without the need for conventional processing overhead.

ARM® Architecture

The ARM® architecture defines two separate exception vectors (i.e., vector addresses), namely, an FIQ vector and an IRQ vector. It is generally recommended that devices that require prompt service be tied to the FIQ vector, whereas devices that have more relaxed time constraints be tied to the IRQ vector. By design, an FIQ interrupt can occur while the machine is servicing an IRQ interrupt. This allows the more time critical device to receive service in a timely manner.

When any of these devices asserts their interrupt line, the instruction stream of the ARM® processor is interrupted. When this interruption occurs, the hardware saves a minimum amount of information and redirects the processor to one of the two vector addresses assigned for the purpose of registering a software handler for the event. It is the responsibility of the handler to save any remaining machine state information and to restore this state information when returning to the interrupted instruction stream. When in the process of saving the machine state, it is important that the processor not be interrupted by a second exception. An interrupt during this process may cause a software error because the machine state may have been partially saved by the first exception. Processing the second exception may cause the software to overwrite this partial data when it saves its machine state. Thus, when processing associated with the first exception completes, the processor may attempt to restore information that was corrupted by the second exception. This potential error is often termed a race condition.

A related aspect of the ARM® core specification is that only the first instruction after receipt of an FIQ or IRQ exception is guaranteed to be serviced without interruption. This is because the ARM® architecture places the FIQ and IRQ vector addresses at immediately successive memory word addresses. The IRQ vector precedes the FIQ vector. This single word address thus leaves room for only one instruction to be executed upon receipt of an IRQ exception before code reserved for the FIQ vector is reached. Since a Branch instruction and Load PC instruction each occupy only one word, there is enough room to place either of these

instructions at a vector address. (This single-word instruction constraint does not apply to FIQ exceptions, since the FIQ vector is the last vector defined by the ARM architecture. Memory space following the FIQ vector address is thus not pre-assigned for vectors, which permits more than one instruction to be placed in memory for use by the FIQ exception handler.)

While the ARM specification effectively guarantees that a Branch or Load PC instruction placed at the IRQ vector address will execute, this guarantees only that the program counter will change. There is still the possibility that an FIQ interrupt may be received while in the midst of saving any remaining machine state information related to the IRQ exception, and thus generate the aforementioned race condition. Because of this situation, current systems are unable to reliably service interrupts from both IRQ and FIQ exceptions without adding processing overhead to detect and recover therefrom.

As discussed in the instant specification, the present invention avoids this race condition and therefore obviates the need for such processing overhead, while it still permits individual devices to be tied to either the IRQ vector or the FIQ vector of the processor. This is accomplished by effectively preventing a second exception from interrupting until critical information associated with the a first exception has been saved. Advantageously, systems that employ this invention may experience significantly faster response times than those using approaches recommended in the ARM Application Note 25 discussed below. The art of record neither recognizes nor addresses this issue, nor does it teach the claimed solution.

As described in the specification, the present invention permits a single handler to process exceptions of distinct interrupt types (e.g., IRQ and FIQ), while avoiding race conditions therebetween. The invention accomplishes this in part by taking advantage of the sequential placement of the processor's IRQ and FIQ vector addresses. Rather than placing a Branch or Load PC instruction at the IRQ address as recommended by ARM® in Note 25, embodiments of the present invention place a Move to Status Register (MSR) instruction there instead. This single instruction effectively locks out the FIQ interrupt. And, since this instruction is placed at the IRQ vector address, it will complete execution before any subsequent exception can interrupt.

The instruction stream thus falls through to the FIQ vector where a conventional Branch or Load PC instruction is placed. The interrupt source (e.g., IRQ or FIQ) may be determined later by examining, for example, the contents of the Current Processor Status Register.

ARM Application Note 25

The ARM® Application Note 25 (Exception Handling on the ARM, ARM DAI 0025E) describes intended interrupt processing for the ARM chip. Section 4 (page 8) of Note 25 describes the process of installing exception handlers. As described, these exception handlers take one or two forms: those using a Branch instruction or those using Load PC instruction. Note 25 then elaborates on the ARM exception processing method by providing exemplary code fragments that illustrate how a programmer might implement exception handlers using the two recommended approaches.

Section 4.1 of Note 25 provides a code fragment written in assembly language, which illustrates the process of loading the vector table. This code fragment loads the IRQ vector with an address of one handler routine, and loads the FIQ vector with the address of another handler routine. The IRQ interrupt handler is referenced as IRQ_Handler and the FIQ interrupt handler is referenced as FIQ Handler. It is thus clear that Note 25 contemplates the use of two discrete interrupt handlers, and does not contemplate the possibility of using a single handler for both exceptions as taught by the instant application.

Accordingly, the claimed approach for loading an IRQ vector and merging interrupt streams upon receipt by a processor is neither disclosed nor suggested by Note 25.

ARM® Application Note 31

ARM® Application Note 31 describes a method to initialize the IRQ and FIQ exception vectors in a system that lacks any interrupt handlers (see page 12, Section 6). This method is intended to be used by developers who are initializing a debugger environment. It is not intended to be an example of how to initialize vectors from a running program. It also does not attempt to describe a process for merging interrupt streams received from IRQ and FIQ devices at IRQ and FIQ vectors.

This Note includes an exemplary code fragment that places an interrupt handler at the FIQ vector address (0X1C). This example uses a Move to Status Register instruction to disable interrupts. However, this instruction is placed at the FIQ vector address rather than at the IRQ vector address as taught by Applicants. Instead, the example places a No-Operation (NOP) instruction at the IRQ address (0X18). Those skilled in the art will recognize that a NOP is routinely used as a space filler, when the address is not needed or is not to be otherwise used. In this example, a NOP was used ostensibly because no devices were to be tied to the IRQ vector, as this example was intended simply for use by developers in a debug environment as discussed above. Although the NOP may inherently allow an IRQ exception to fall through to the FIQ address, it would do so without ensuring that the FIQ interrupt is disabled first.

As such, Note 31 should not be viewed as disclosing a method for merging IRQ and FIQ interrupt streams as taught by the Applicants. Moreover, it is evident that Note 31 fails to identify or address the problem of race conditions that would be associated with any merged interrupt streams. Rather, this exemplary code fragment appears to simply be directed towards teaching use of the free address space below the FIQ vector address.

It is thus evident that Note 31 does not anticipate use of its exemplary approach in a running program, but instead, simply teaches its use by developers in a debug environment, e.g., where race conditions can be monitored and/or avoided. Note 31 should therefore be viewed as failing to teach the merger of discrete interrupt streams upon receipt by a processor as taught by the instant application.

Bhagat

The Bhagat patent discloses a hardware device that is external to the ARM® processor 102. This hardware device is referred to as an interrupt controller 106. As shown in Fig. 1, controller 106 receives interrupts from several external devices (108-126) upstream of the ARM® processor 102, and then transmits either an FIQ interrupt 128 or an IRQ interrupt 130 to the processor. Bhagat does not teach or address the difficulty of merging the FIQ and IRQ interrupt streams once they are received at the FIQ and IRQ vectors of the ARM® chip. Rather, Bhagat's processor 102 appears to handle the FIQ and IRQ interrupts in an entirely conventional

manner. Indeed, in paragraph 0023, Bhagat discusses the conventional use of these vectors, stating that the IRQ vector may be used for "a branch instruction to the full exception handler...".

Bhagat discusses a 'global disable' (see claim 5). However, this refers to a feature of Bhagat's interrupt controller 106, not to a feature of the ARM® instruction set. Furthermore, Bhagat's reference in claim 6 to a "common part of an interrupt service routine" is not a reference to a common interrupt service routine as disclosed by Applicants. Rather, this "common part" language simply refers to a portion of Bhagat's service routine and is entirely distinct from Applicants' teaching of merging two interrupt streams (IRQ, FIQ) into a single service routine. Moreover, Bhagat neither anticipates nor suggests placing an instruction at the IRQ vector address that would disable interrupts for the purpose of alleviating race conditions in interrupt streams that are merged upon receipt by the processor.

Applicants therefore submit that neither Bhagat, Notes 25 and 31, nor any of the other cited references, teach the concept of merging interrupt streams downstream of the processor's interrupt (e.g., IRQ and FIQ) vectors. Rather, one or more of these references describe servicing interrupts from either vector independently. None of these references address the problem of race conditions between interrupt streams. In addition, none of these references addresses the pitfalls of merging interrupt streams from two sources at the processor level without making provision for such race conditions. Any description in these references of enabling and disabling interrupts either refers to devices separate and operationally upstream from the IRQ and FIQ vectors (Bhagat), or is implemented only in conjunction with multi-instruction FIQ interrupt service routines.

Applicants' invention describes a process for advantageously merging and eliminating race conditions between first and second type (e.g., IRQ and FIQ) interrupts for improved processing speed, efficiency, and accuracy, while preserving the ability to use and distinguish between both types of interrupts. Applicants provide this functionality in part by disabling the first and second type interrupts upon receipt of a first type interrupt using a single instruction inserted at the first type vector address. Processing of the received first type interrupt then falls to second type vector address which branches to a common interrupt handler. This interrupt is

then processed without interruption. The advantages provided by Applicants' approach are neither present nor taught by the art of record alone or in combination. The claimed invention should thus be viewed as patentably distinct over the cited references alone or in combination.

In light of the foregoing, Applicants respectfully submit that one of ordinary skill in the art would not have looked to the teachings of Bhagat, which present a controller disposed upstream of the IRQ and FIQ vectors of an ARM processor, and which feeds separate IRQ and FIQ interrupt streams to the IRQ and FIQ vector addresses, to somehow merge these streams downstream of the vector addresses. At least one, if not all, of the references shows no recognition of the race condition problem faced by Applicants. As such, the combination of the cited references is not sufficiently pertinent to the particular problem faced by Applicants as to reasonably suggest Applicants' claimed invention to those skilled in the art. Absent such a teaching, suggestion or incentive supporting the combination, one skilled in the art would not have been motivated to combine Bhagat with any of the other cited references to produce the subject invention.

Even if properly combined, the result would not meet the limitations of the independent claims, since the interrupt streams would be merged upstream, rather than downstream of the processor's interrupt vectors as claimed. Further, any such combination would omit the use of a single instruction placed at the first (e.g., IRQ) vector that disables subsequent interrupts.

For each of the foregoing alternate reasons, applicants respectfully request reconsideration and allowance of the amended claims presented herewith. The dependent claims are believed allowable for the same reasons as the independent claims from which they depend, as well as for their own additional limitations.

CONCLUSION

Applicants submit that all of the stated grounds of objection and rejection have been properly traversed, accommodated, or rendered moot. This application is now believed to be in condition for allowance, and such action at an early date is respectfully requested. However, if any matters remain unresolved, the Examiner is encouraged to contact the undersigned by telephone.

In the unlikely event that the transmittal letter is separated from this document and the Patent Office determines that an extension and/or other relief is required, applicant petitions for any required relief including extensions of time and authorizes the Assistant Commissioner to charge the cost of such petitions and/or other fees due in connection with the filing of this document to **Deposit Account No. 50-0374** referencing docket no. 2000.021/1109.005. However, the Assistant Commissioner is not authorized to charge the cost of the issue fee to the Deposit Account.

Respectfully submitted,

Richard L. Sampson
Attorney for Applicants
Registration No. 37,231

Dated: June 30, 2004

SAMPSON & ASSOCIATES, P.C.
50 Congress Street
Suite 519
Boston, MA 02109
Telephone: (617) 557-2900
Facsimile: (617) 557-0077